

(1)	121	Declarations and Equates
(1)	196	Transfer Vector and Service Definitions
(1)	235	Change Mode Dispatcher Vector Block
(1)	283	Kernel Mode Dispatcher
(1)	340	Intercept system service \$SSI_USSK user system service
(2)	358	Intercept system service \$SSI_USSU user system service


```
0000 1      .TITLE SSI_SYS_DISP - user system service dispatcher
0000 2      .IDENT 'V04-000'
0000 3
0000 4      *****
0000 5      *
0000 6      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 7      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 8      *  ALL RIGHTS RESERVED.
0000 9      *
0000 10     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 11     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 12     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 13     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 14     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 15     *  TRANSFERRED.
0000 16     *
0000 17     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 18     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 19     *  CORPORATION.
0000 20     *
0000 21     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 22     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 23     *
0000 24     *  *****
0000 25     *****
0000 26
0000 27
0000 28     ++
0000 29     Abstract:
0000 30         This module is the interface routine for privileged shareable
0000 31         image DBGSSISHR.EXE. DBGSSISHR defines the dispatcher and
0000 32         calls the routines to set up system service interception.
0000 33
0000 34     Modified by:
0000 35         Ping Sager
0000 36
0000 37     Origin:
0000 38         SYSSYSROOT:[SYSHLP.EXAMPLES]USSDISP.MAR
0000 39         This module contains an example dispatcher for user written
0000 40         system services along with several sample services and a user
0000 41         rundown example. It is a template intend to serve as the starting
0000 42         point for implementing a privileged shareable image containing your
0000 43         own services. When used as a template, the definitions and code
0000 44         for the sample services should be removed.
0000 45
0000 46     Overview:
0000 47         User written system services are contained in privileged shareable
0000 48         images that are linked into user program images in exactly the
0000 49         same fashion as any shareable image. The creation and installation
0000 50         of a privileged, shareable image is slightly different from that
0000 51         of an ordinary shareable image. These differences are:
0000 52
0000 53         1. A vector defining the entry points and providing other
0000 54            control information to the image activator. This vector
0000 55            is a the lowest address in an image section with the VEC
0000 56            attribute.
0000 57
```


0000 58 :
0000 59 :
0000 60 :
0000 61 :
0000 62 :
0000 63 :
0000 64 :
0000 65 :
0000 66 :
0000 67 :
0000 68 :
0000 69 :
0000 70 :
0000 71 :
0000 72 :
0000 73 :
0000 74 :
0000 75 :
0000 76 :
0000 77 :
0000 78 :
0000 79 :
0000 80 :
0000 81 :
0000 82 :
0000 83 :
0000 84 :
0000 85 :
0000 86 :
0000 87 :
0000 88 :
0000 89 :
0000 90 :
0000 91 :
0000 92 :
0000 93 :
0000 94 :
0000 95 :
0000 96 :
0000 97 :
0000 98 :
0000 99 :
0000 100 :
0000 101 :
0000 102 :
0000 103 :
0000 104 :
0000 105 :
0000 106 :
0000 107 :
0000 108 :
0000 109 :
0000 110 :
0000 111 :
0000 112 :
0000 113 :
0000 114 :

2. The shareable image is linked with the /PROTECT option that marks all of the image sections so that they will be protected and given EXEC mode ownership by the image activator.
3. The shareable image MUST be installed /SHARE /PROTECT with the INSTALL utility in order for the image activator to connect the privileged shareable image to the change mode dispatchers.

A privileged shareable image implementing user written system services is comprised of the following major components:

1. A transfer vector containing all of the entry points and collecting them at the lowest virtual address in the shareable image. This formalism enables revision of the shareable image without necessitating the relinking of images that use it.
2. A Privileged Library Vector in a PSECT with the VEC attribute that describes the entry points for dispatching EXEC and KERNEL mode services along with validation information.
3. A dispatcher for kernel mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize a kernel mode service request.
4. A dispatcher for executive mode services. This code will be called by the VMS change mode dispatcher when it fails to recognize an executive mode service request.
5. Service routines to perform the various services.

The first four components are contained in this template and are most easily implemented in MACRO, while the service routines can be implemented in BLISS or MACRO. Other languages may be usable but are not recommended -- particularly if they require runtime support routines or are extravagant in their use of stack or are unable to generate PIC code.

This example is position-independent (PIC) and it is good practice to implement shareable images this way whenever possible.

--
Link Command File Example:

```
$!  
$! Command file to link User System Service example.  
$!  
$ LINK/PROTECT/NOSYSSHR/SHARE=USS/MAP=USS/FULL SYSS$INPUT/OPTIONS  
$!  
Options file for the link of User System Service example.  
SYSS$SYSTEM:SYS.STB/SELECTIVE  
$!  
Create a separate cluster for the transfer vector.
```


SSI SYS-DISP
V04=000

- user system service dispatcher D 2

15-SEP-1984 23:40:11 VAX/VMS Macro V04-00
5-SEP-1984 00:07:24 [DEBUG.SRC]SSIDISP.MAR;1

Page 3
(1)

0000	115	:	CLUSTER=TRANSFER_VECTOR,,,SYSSDISK:[]USSDISP
0000	116	:	!
0000	117	:	GSMATCH=LEQUAL,1,1
0000	118	:	
0000	119	:--	


```
0000 121 .SBTTL Declarations and Equates
0000 122 :
0000 123 : Include Files
0000 124 :
0000 125 :
0000 126 .EXTERNAL SSIK_START
0000 127 .EXTERNAL SSIU_START
0000 128
0000 129 $PRVDEF
0000 130 $PSLDEF
0000 131 $OPDEF
0000 132
0000 133
0000 134 :
0000 135 : Macro Definitions
0000 136 :
0000 137 DEFINE_SERVICE - A macro to make the appropriate entries in several
0000 138 different PSECTs required to define an EXEC or KERNEL
0000 139 mode service. These include the transfer vector,
0000 140 the case table for dispatching, and a table containing
0000 141 the number of required arguments.
0000 142
0000 143 DEFINE_SERVICE Name,Entry_Point,Number_of_Arguments,Mode
0000 144 :
0000 145 .MACRO DEFINE_SERVICE,NAME,ENTRY,NARG=0,MODE=KERNEL
0000 146 .PSECT $$$TRANSFER_VECTOR,PAGE,NOWRT,EXE,PIC
0000 147 .ALIGN QUAD ; Align entry points for speed and style
0000 148 .TRANSFER NAME ; Define name as universal symbol for entry
0000 149 NAME:: ; Use entry mask defined in main routine
0000 150 .MASK ENTRY
0000 151 .IF IDN MODE,KERNEL
0000 152 CHMK #<KCODE_BASE+KERNEL_COUNTER> ; Change to kernel mode and execute
0000 153 RET ; Return
0000 154 KERNEL_COUNTER=KERNEL_COUNTER+1 ; Advance counter
0000 155
0000 156 .PSECT KERNEL_NARG,BYTE,NOWRT,EXE,PIC
0000 157 .BYTE NARG ; Define number of required arguments
0000 158
0000 159 .PSECT USER_KERNEL_DISP1,BYTE,NOWRT,EXE,PIC
0000 160 .WORD 2+ENTRY-KCASE_BASE ; Make entry in kernel mode CASE table
0000 161 .ENDC
0000 162
0000 163 .IF IDN MODE,USER
0000 164 JMP ENTRY+2
0000 165
0000 166 .PSECT USER_NARG,BYTE,NOWRT,EXE,PIC
0000 167 .BYTE NARG ; Define number of required arguments
0000 168 .ENDC
0000 169
0000 170 .ENDM DEFINE_SERVICE
0000 171
0000 172 :
0000 173 : Equated Symbols
0000 174 :
0000 175 :
0000 176
0000 177 $PHDDEF ; Define process header offsets
```



```

0000 178      $PLVDEF      ; Define PLV offsets and values
0000 179      $SSDEF      ; Define system status code
0000 180      ;
0000 181      ; Initialize counters for change mode dispatching codes
0000 182      ;
00000000 0000 183  KERNEL_COUNTER=0      ; Kernel code counter
0000 184      ;
0000 185      ;
0000 186      ; Own Storage
0000 187      ;
00000000 0000 188      .PSECT  KERNEL_NARG,BYTE,NOWRT,EXE,PIC
0000 189  KERNEL_NARG:      ; Base of byte table containing the
0000 190      ; number of required arguments.
0000 191      ;
00000000 0000 192      .PSECT  USER_NARG,BYTE,NOWRT,EXE,PIC
0000 193  USER_NARG:
0000 194

```



```
0000 196 .SBTTL Transfer Vector and Service Definitions
0000 197 :++
0000 198 : The use of transfer vectors to effect entry to the user written system services
0000 199 : enables some updating of the shareable image containing them without necessitating
0000 200 : a re-link of all programs that call them. The PSECT containing the transfer
0000 201 : vector will be positioned at the lowest virtual address in the shareable image
0000 202 : and so long as the transfer vector is not re-ordered, programs linked with
0000 203 : one version of the shareable image will continue to work with the next.
0000 204 :
0000 205 : Thus as additional services are added to a privileged shareable image, their
0000 206 : definitions should be added to the end of the following list to ensure that
0000 207 : programs using previous versions of it will not need to be re-linked.
0000 208 : To completely avoid relinking existing programs the size of the privileged
0000 209 : shareable image must not change so some padding will be required to provide
0000 210 : the opportunity for future growth.
0000 211 :--
0000 212 DEFINE_SERVICE SSI_USSK,INT_SSI_USSK,1,KERNEL
0002 213 ; Service to setup intercept
0002 214 ; system service
0002 215
0002 216 DEFINE_SERVICE SSI_USSU,INT_SSI_USSU,4,USER
0001 217 ; Set up to use intercept
0001 218 ; system service in user mode
0001 219
0001 220 :
0001 221 : The base values used to generate the dispatching codes should be negative for
0001 222 : user services and must be chosen to avoid overlap with any other privileged
0001 223 : shareable images that will be used concurrently. Their definition is
0001 224 : deferred to this point in the assembly to cause their use in the preceding
0001 225 : macro calls to be forward references that guarantee the size of the change
0001 226 : mode instructions to be four bytes. This satisfies an assumption that is
0001 227 : made by for services that have to wait and be retried. The PC for retrying
0001 228 : the change mode instruction that invokes the service is assumed to be 4 bytes
0001 229 : less than that saved in the change mode exception frame. Of course, the particula
0001 230 : service routine determines whether this is possible.
0001 231 :
000040B8 0001 232 KCODE_BASE=16568 ; Base CHMK code value for these services
0001 233
```


0001 235 .SBTTL Change Mode Dispatcher Vector Block

0001 236 :++
0001 237 : This vector is used by the image activator to connect the privileged shareable
0001 238 : image to the VMS change mode dispatcher. The offsets in the vector are self-
0001 239 : relative to enable the construction of position independent images. The system
0001 240 : version number will be used by the image activator to verify that this shareable
0001 241 : image was linked with the symbol table for the current system.
0001 242 :
0001 243 :
0001 244 :
0001 245 :
0001 246 :
0001 247 :
0001 248 :
0001 249 :
0001 250 :
0001 251 :
0001 252 :
0001 253 :
0001 254 :
0001 255 :
0001 256 :
0001 257 :
0001 258 :
0001 259 :
0001 260 :
0001 261 :
0001 262 :
0001 263 :
0001 264 :
0001 265 :
0001 266 :
0001 267 :
0001 268 :
0001 269 :
0001 270 :
0001 271 :
00000000 272 :
00000000 273 :
00000001 0000 274 :
00000000 0004 275 :
00000003 0008 276 :
00000000 000C 277 :
0000007C 0010 278 :
00000000 0014 279 :
00000000 0018 280 :
00000000 001C 281 :

Change Mode Vector Format

Vector Type Code (PLVSC_TYP_CM0D)	PLVSL_TYPE
System Version Number (SYSSK_VERSION)	PLVSL_VERSION
Kernel Mode Dispatcher Offset	PLVSL_KERNEL
Exec Mode Entry Offset	PLVSL_EXEC
User Rundown Service Offset	PLVSL_USRUNDWN
Reserved	
RMS Dispatcher Offset	PLVSL_RMS
Address Check	PLVSL_CHECK

.PSECT USER_SERVICES,PAGE,VEC,PIC,NOWRT,EXE

.LONG	PLVSC_TYP_CM0D	: Set type of vector to change mode dispatch
.LONG	SYSSK_VERSION	: Identify system version
.LONG	KERNEC_DISPATCH-	: Offset to kernel mode dispatcher
.LONG	0	: No executive mode dispatcher
.LONG	CLEANUP_SSV-	: User rundown service to restore SSV
.LONG	0	: Reserved
.LONG	0	: No RMS dispatcher
.LONG	0	: Address check - PIC image


```
0020 283 .SBTTL Kernel Mode Dispatcher
0020 284 :++
0020 285 : Input Parameters:
0020 286 :
0020 287 : (SP) - Return address if bad change mode value
0020 288 :
0020 289 : R0 - Change mode argument value.
0020 290 :
0020 291 : R4 - Current PCB Address. (Therefore R4 must be specified in all
0020 292 : register save masks for kernel routines.)
0020 293 :
0020 294 : AP - Argument pointer existing when the change
0020 295 : mode instruction was executed.
0020 296 :
0020 297 : FP - Address of minimal call frame to exit
0020 298 : the change mode dispatcher and return to
0020 299 : the original mode.
0020 300 :--
0000 0000 301 .PSECT USER_KERNEL_DISP0,BYTE,NOWRT,EXE,PIC
50 0C 3C 0000 302 KACCVIO: ; Kernel access violation
04 0003 303 MOVZWL #SS$_ACCVIO,R0 ; Set access violation status code
0004 304 RET ; and return
50 0114 BF 3C 0004 305 KINSFARG: ; Kernel insufficient arguments.
04 0009 306 MOVZWL #SS$_INSFARG,R0 ; Set status code and
05 000A 307 RET ; return
000B 308 KNOTME: RSB ; RSB to forward request
000B 309
000B 310 KERNEL_DISPATCH: ; Entry to dispatcher
51 BF48 C0 9E 000B 311 MOVAB W^=KCODE_BASE(R0),R1 ; Normalize dispatch code value
01 51 B1 19 0010 312 BLSS KNOTME ; Branch if code value too low
F3 1E 0012 313 CMPW R1,#KERNEL_COUNTER ; Check high limit
0015 314 BGEQU KNOTME ; Branch if out of range
0017 315
0017 316 : The dispatch code has now been verified as being handled by this dispatcher,
0017 317 : now the argument list will be probed and the required number of arguments
0017 318 : verified.
0017 319
0017 320 MOVZBL W^KERNEL_NARG[R1],R1 ; Get required argument count
51 0000'CF41 9A 0017 321 MOVAL @#4[R1],R1 ; Compute byte count including arg count
00000004 9F41 DE 001D 322 IFNORD R1,(AP),KACCVIO ; Branch if arglist not readable
BF48'CF40 6C 91 0025 323 CMPB (AP),W^KERNEL_NARG-KCODE_BASE[R0] ; Check for required number
D1 1F 002B 324 BLSSU KINSFARG ; of arguments
50 AF 0031 325 CASEW R0,- ; Case on change mode
0035 326 - ; argument value
0035 327 #KCODE_BASE,- ; Base value
00 40B8 BF 0035 328 #<KERNEL_COUNTER-1> ; Limit value (number of entries)
0039 329 KCASE_BASE: ; Case table base address for DEFINE_SERVICE
0039 330
0039 331 : Case table entries are made in the PSECT USER_KERNEL_DISP1 by
0039 332 : invocations of the DEFINE_SERVICE macro. The three PSECTS,
0039 333 : USER_KERNEL_DISP0,1,2 will be abutted in lexical order at link-time.
0039 334
0000 0000 335 .PSECT USER_KERNEL_DISP2,BYTE,NOWRT,EXE,PIC
0000 336 BUG_CHECK IVSSRVQRST,FATAL ; Since the change mode code is validated
0004 337 ; above, we should never get here
0004 338
```


SSI_SYS_DISP
V04-000

J 2
- user system service dispatcher
Intercept system service \$SSI_USSK user 15-SEP-1984 23:40:11 VAX/VMS Macro V04-00
5-SEP-1984 00:07:24 [DEBUG.SRC]SSIDISP.MAR;1

Page 9
(1)

```
0004 340 .SBTTL Intercept system service $SSI_USSK user system service
0004 341 :++
0004 342 : Functional Description:
0004 343 : This routine sets up the interface to the real routines which
0004 344 : perform the real work.
0004 345 : Input Parameters:
0004 346 : R4 - Address of current PCB
0004 347 :
0004 348 : Output Parameters:
0004 349 : R0 - Completion Status Code
0004 350 : --
0004 351 :
0004 352 .ENTRY INT_SSI_USSK,^M<R4>
0006 353 PUSH 4,AP
0009 354 CALLS #1,G^SSIK_START ; Invoke routine in privilege library
0010 355 RET
```

00000000'GF 04 AC 0010 DD 0006 FB 0009 04 0010


```
0011 358 .SBTTL Intercept system service $SSI_USSU user system service
0011 359
0011 360 :++ Functional Description:
0011 361 : This routine sets up the interface to the real routines which
0011 362 : perform the real work.
0011 363 : Input Parameters:
0011 364 : See Argument List
0011 365 : Output Parameters:
0011 366 : R0 - Completion Status Code
0011 367 :--
0011 368
0011 369 .ENTRY INT_SSI_USSU,*M<>
0013 370 CMPL #4,(AP) : Require to have 4 arguments
0016 371 BNEQ UINSFARG : Not 4
0018 372 MOVZBL (AP),R1 : Get required argument count
001B 373 MOVAL @#4(R1),R1 : Compute byte count including arg count
0023 374 IFNORD R1,(AP),UACCVIO : Branch if arglist not readable
0029 375 BRB START : Ok
002B 376
002B 377 UACCVIO: : User access violation
002B 378 MOVZWL #SS$_ACCVIO,R0 : Set access violation status code
002E 379 RET : and return
002F 380
002F 381 UINSFARG: : Kernel insufficient arguments
002F 382 MOVZWL #SS$_INSFARG,R0 : Set access violation status code
0034 383 RET : and return
0035 384
0035 385 START:
0035 386 PUSHL 16(AP) : SAVE_MASK, the running state of
0038 387 : all the user declared routines
0038 388 : in different priorities before
0038 389 : changing by USER_MASK, this value
0038 390 : is returned to the caller.
0038 391 PUSHL 12(AP) : USER_ID, used for checking purpose
003B 392 : in SSI_START, this value is
003B 393 : returned to the caller.
003B 394 PUSHL 8(AP) : USER_ADDR, user declared routine,
003E 395 : to be called at interception time.
003E 396 PUSHL 4(AP) : USER_SETUP_FLAGS has 4 bytes, (one byte
0041 397 : each for SETUP_FLAG, USER_PRIO,
0041 398 : USER_MASK & USER_MODE). User
0041 399 : sets up the status for declared
0041 400 : routine.
0041 401
0041 402 PROBEW #0,#4,@8(SP) : Writability of USER_ID
0046 403 BNEQ 2$
0048 404 BRW UACCVIO
004B 405
004B 406 2$: PROBEW #0,#4,@12(SP) : Writability of SAVE_MASK
0050 407 BNEQ 4$
0052 408 BRW UACCVIO
0055 409
0055 410 : Check SETUP_FLAG (0-1)
0055 411 4$: CMPB 0(SP),#1 : Check for legal setup flag
0058 412 BLEQU 8$ : 0 - Disable, 1 - Enable
005A 413 BRW 100$
005D 414
```

6C 04 0000 0013 370
51 00000004 9F41 DE 001B 373
0A 11 0029 375
50 0C 3C 002B 377
04 002E 379
50 0114 8F 3C 002F 381
04 0034 383
10 AC DD 0035 385
0C AC DD 0038 391
08 AC DD 003B 394
04 AC DD 003E 396
0B BE 04 00 0D 0041 401
03 12 0046 403
FFE0 31 0048 404
0C BE 04 00 0D 004B 405
03 12 0050 407
FFD6 31 0052 408
01 6E 91 0055 410
03 1B 0058 412
002B 31 005A 413
005D 414


```
01 01 AE 91 005D 415 : Check USER_Prio (1-4)
01 01 AE 91 005D 416 8$: CMPB 1(SP),#1 ; Check for legal priority
01 01 AE 91 0061 417 BLSS 10$ ; 1, 2, 3, 4
04 01 AE 91 0063 418 CMPB 1(SP),#4 ; 1, 2 - User program in user mode
01 02 14 0067 419 BGTR 10$ ; 3 - DEBUG in user mode
01 03 11 0069 420 BRB 12$ ;
01 03 31 006B 421 10$: ; 4 - Super DEBUG in user mode
01 03 31 006B 422 BRW 100$
01 03 31 006E 423
01 03 31 006E 424
01 03 31 006E 425 : We don't check USER_MASK here, for we only take one bit value for
01 03 31 006E 426 : prio. 1, 2 bits for prio. 2, 3 bits for prio. 3, and 4 bits for prio. 4.
01 03 31 006E 427 : in SSI_START. Note: this is the way we return mask value back to user.
01 03 31 006E 428 :
01 03 31 006E 429
01 03 31 006E 430 : Check USER_MODE (3)
01 03 31 006E 431 12$: CMPB 3(SP), #PSL$C_USER ; Check for legal mode
01 03 31 0072 432 BEQL 14$ ; Only accept user mode
01 03 31 0074 433 BRW 100$
01 03 31 0077 434
01 03 31 0077 435 : Check USER_ID (0 OR 1-16)
01 03 31 0077 436 14$: CMPL 28(SP),#16 ; Check for legal USER_ID value
01 03 31 007B 437 BLEQU 20$ ; range: 0 to 16
01 03 31 007D 438 BRW 100$
01 03 31 0080 439
01 03 31 0080 440 : All checking is done, ready to call the routines to do the real work.
01 03 31 0080 441 :
01 03 31 0080 442 20$:
01 03 31 0080 443 CALLS #4,G^SSIU_START
01 03 31 0087 444 RET
01 03 31 0088 445
01 03 31 0088 446 100$:
01 03 31 0088 447 MOVL #0,R0
01 03 31 008B 448 RET
01 03 31 008C 449
01 03 31 008C 450 : This routine is declared as the user rundown service. It gets called
01 03 31 008C 451 : at image rundown. Its purpose is to restore the system service vector
01 03 31 008C 452 : to its original state. This is done by calling the COPY_SAVED_SSV
01 03 31 008C 453 : routine in module RESETSSV. Note - this routine does not want to
01 03 31 008C 454 : restore the system service vector twice. It checks for this both
01 03 31 008C 455 : by checking to see whether there is are JSBs in the system service
01 03 31 008C 456 : vector and also checking a global flag that says whether we have
01 03 31 008C 457 : changed the system service vector.
01 03 31 008C 458 :
01 03 31 008C 459 CLEANUP_SSV::
01 03 31 008C 460 -CMPB @#SYSS$RUNDWN+2,#OPS_JSB ; Has the system service vector
01 03 31 0093 461 ; been changed? (If so, we
01 03 31 0093 462 ; will have put a JSB instruction
01 03 31 0093 463 ; at SYSS$anything+2).
01 03 31 0093 464 BNEQ 10$ ; No, skip call which restores SSV
01 03 31 0095 465 TSTL SSV_MUNGED_FLAG ; Has the system service vector
01 03 31 009B 466 ; been changed?
01 03 31 009B 467 BEQL 10$ ; No, skip call which restores SSV
01 03 31 009D 468 CALLS #0,COPY_SAVED_SSV ; Restore system service vector
01 03 31 00A4 469 10$: RSB
01 03 31 00A5 470
01 03 31 00A5 471 .END
```


SSI SYS DISP
Symbol table

- user system service dispatcher M 2

15-SEP-1984 23:40:11 VAX/VMS Macro V04-00
5-SEP-1984 00:07:24 [DEBUG.SRC]SSIDISP.MAR;1

Page 12
(2)

BUGS_IVSSRVQST	*****	X	08
CLEANUP_SSV	0000008C	RG	08
COPY_SAVED_SSV	*****	X	08
INT_SSI_USSK	00000004	RG	08
INT_SSI_USSU	00000011	RG	08
KACCVIO	00000000	R	07
KCASE_BASE	00000039	R	07
KCODE_BASE	= 000040B8		
KERNEL_COUNTER	= 00000001		
KERNEL_DISPATCH	0000000B	RG	07
KERNEL_NARG	00000000	R	02
KINSFARG	00000004	R	07
KNOTME	0000000A	R	07
OPS_JSB	= 00000016		
PLVSC_TYP_CM0D	= 00000001		
PSLSC_USER	= 00000003		
SSS_ACCVIO	= 0000000C		
SSS_INSFARG	= 00000114		
SSIR_START	*****	X	00
SSIU_START	*****	X	00
SSI_USSK	00000000	RG	04
SSI_USSU	00000008	RG	04
SSV_MUNGED_FLAG	*****	X	08
START	00000035	R	08
SYSSK_VERSION	*****	X	06
SYSSR0NDWN	*****	X	08
UACCVIO	0000002B	R	08
UINSFARG	0000002F	R	08
USER_NARG	00000000	R	03

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
KERNEL_NARG	00000001 (1.)	02 (2.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
USER_NARG	00000001 (1.)	03 (3.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
\$\$\$TRANSFER_VECTOR	00000010 (16.)	04 (4.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC PAGE
USER_KERNEL_DISP1	00000002 (2.)	05 (5.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
USER_SERVICES	00000020 (32.)	06 (6.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT VEC PAGE
USER_KERNEL_DISP0	00000039 (57.)	07 (7.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE
USER_KERNEL_DISP2	000000A5 (165.)	08 (8.)	PIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	10	00:00:00.07	00:00:01.68
Command processing	72	00:00:00.74	00:00:03.85
Pass 1	293	00:00:09.22	00:00:26.83
Symbol table sort	0	00:00:01.64	00:00:05.20
Pass 2	110	00:00:01.92	00:00:06.84

Symbol table output	5	00:00:00.05	00:00:00.05
Psect synopsis output	3	00:00:00.04	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	495	00:00:13.69	00:00:44.51

The working set limit was 1350 pages.

53089 bytes (104 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1037 non-local and 9 local symbols.
471 source lines were read in Pass 1, producing 31 object records in Pass 2.
17 pages of virtual memory were used to define 15 macros.

! Macro library statistics !

Macro library name

Macros defined

\$255\$DUA28:[SYS.OBJ]LIB.MLB;1
\$255\$DUA28:[SYSLIB]STARLET.MLB;2
TOTALS (all libraries)

3
8
11

1104 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:SSIDISP/OBJ=OBJ\$:SSIDISP MSRC\$:SSIDISP/UPDATE=(ENH\$:SSIDISP)+EXECML\$/LIB

0101 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

